Contents

HTTP Traffic

4 minute read ✓ page test

This task shows you how to set up Istio authorization policy of

`ALLOW` action for HTTP traffic in an Istio mesh.

Before you begin

Before you begin this task, do the following:

- Read the Istio authorization concepts.
- Follow the Istio installation guide to install Istio with mutual TLS enabled.
- Deploy the Bookinfo sample application.

After deploying the Bookinfo application, go to the Bookinfo product page at `http://$GATEWAY_URL/productpage`. On the product page, you can see the following sections:

- **Book Details** on the lower left side, which includes: book type, number of pages, publisher, etc.
- **Book Reviews** on the lower right of the page.

When you refresh the page, the app shows different versions of reviews in the product page. The app presents the reviews in a round robin style: red stars, black stars, or no stars.

> If you don't see the expected output in the browser as you follow the task, retry in a few more seconds because some delay is possible due to caching and

other propagation overhead.

⚠ This task requires mutual TLS enabled because the following examples use principal and namespace in the policies.

Configure access control for workloads using HTTP traffic

Using Istio, you can easily setup access control for workloads in your mesh. This task shows you how to set up access control using Istio authorization. First, you configure a simple `allow-nothing` policy that rejects all requests to the workload, and then grant more access to the workload gradually and incrementally.

1. Run the following command to create a `allow-nothing` policy in the `default` namespace. The policy doesn't have a `selector` field, which applies the policy to every workload in the `default` namespace. The `spec:` field of the policy has the empty value `{}`. That value means that no traffic is permitted, effectively denying all requests.

```
$ kubectl apply -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: allow-nothing
  namespace: default
spec:
  {}
EOF
```

Point your browser at the Bookinfo `productpage` (`http://$GATEWAY_URL/productpage`). You should see `"RBAC: access denied"`. The error shows that the configured `deny-all` policy is working as intended, and Istio doesn’t have any rules that allow any access to workloads in the mesh.

2. Run the following command to create a `productpage-viewer` policy to allow access with `GET` method to the `productpage`

workload. The policy does not set the `from` field in the `rules` which means all sources are allowed, effectively allowing all users and workloads:

```
$ kubectl apply -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: "productpage-viewer"
  namespace: default
spec:
  selector:
    matchLabels:
      app: productpage
  action: ALLOW
  rules:
  - to:
    - operation:
        methods: ["GET"]
EOF
```

   Point your browser at the Bookinfo `productpage` (`http://$GATEWAY_URL/productpage`). Now you should see the “Bookinfo Sample” page. However, you can see the following errors on the page:

   - `Error fetching product details`
   - `Error fetching product reviews` on the page.

   These errors are expected because we have not granted the `productpage` workload access to the `details` and `reviews` workloads. Next, you need to configure a policy to grant access to those workloads.

3. Run the following command to create the `details-viewer` policy to allow the `productpage` workload, which issues requests using the `cluster.local/ns/default/sa/bookinfo-productpage` service account, to access the `details` workload

through `GET` methods:

```
$ kubectl apply -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: "details-viewer"
  namespace: default
spec:
  selector:
    matchLabels:
      app: details
  action: ALLOW
  rules:
  - from:
    - source:
        principals: ["cluster.local/ns/default/sa/bookinfo-productpage
"]
    to:
    - operation:
        methods: ["GET"]
EOF
```

4. Run the following command to create a policy `reviews-viewer` to allow the `productpage` workload, which issues requests using the `cluster.local/ns/default/sa/bookinfo-productpage` service account, to access the `reviews` workload through `GET` methods:

```
$ kubectl apply -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: "reviews-viewer"
  namespace: default
spec:
  selector:
    matchLabels:
      app: reviews
  action: ALLOW
  rules:
  - from:
    - source:
        principals: ["cluster.local/ns/default/sa/bookinfo-productpage
"]
    to:
    - operation:
        methods: ["GET"]
EOF
```

Point your browser at the Bookinfo `productpage` (`http://$GATEWAY_URL/productpage`). Now, you should see the "Bookinfo Sample" page with "Book Details" on the lower left part, and "Book Reviews" on the lower right part. However, in the "Book Reviews" section, there is an error `Ratings service currently unavailable`.

This is because the `reviews` workload doesn't have permission to access the `ratings` workload. To fix this issue, you need to grant the `reviews` workload access to the `ratings` workload. Next, we configure a policy to grant the `reviews` workload that access.

5. Run the following command to create the `ratings-viewer` policy to allow the `reviews` workload, which issues requests using the `cluster.local/ns/default/sa/bookinfo-reviews`

service account, to access the `ratings` workload through `GET` methods:

```
$ kubectl apply -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: "ratings-viewer"
  namespace: default
spec:
  selector:
    matchLabels:
      app: ratings
  action: ALLOW
  rules:
  - from:
    - source:
        principals: ["cluster.local/ns/default/sa/bookinfo-reviews"]
    to:
    - operation:
        methods: ["GET"]
EOF
```

Point your browser at the Bookinfo `productpage` (`http://$GATEWAY_URL/productpage`). You should see the “black” and “red” ratings in the “Book Reviews” section.

Congratulations! You successfully applied authorization policy to enforce access control for workloads using HTTP traffic.

Clean up

1. Remove all authorization policies from your configuration:

```
$ kubectl delete authorizationpolicy.security.istio.io/allow-nothing
$ kubectl delete authorizationpolicy.security.istio.io/productpage-viewer
$ kubectl delete authorizationpolicy.security.istio.io/details-viewer
$ kubectl delete authorizationpolicy.security.istio.io/reviews-viewer
$ kubectl delete authorizationpolicy.security.istio.io/ratings-viewer
```